Problem 6 (5 points)

In [28]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.linear_model import LogisticRegression
```

Multinomial Classification in SciKit-Learn

Load Dataset

(Don't edit this)

- (x,y) values are stored in rows of `xy`
- class values are in `c`

In [29]:
```
x = np.array([7.4881350392732475,16.351893663724194,22.427633760716436,29.048831
y = np.array([0.11120957227224215,0.1116933996874757,0.14437480785146242,0.11818
xy = np.vstack([x,y]).T
c = np.array([0,2,2,2,2,2,0,2,2,2,2,2,0,0,2,0,1,2,0,0,1,1,1,2,0,1,0,1,1,1,0,0,1,
```

Logistic Regression

SciKit-Learn's Logistic Regression model will perform multinomial classification automatically.

Create an sklearn `LogisticRegression()` class and train this model on the dataset

Details about how to use this are here: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

In [32]:

```
from sklearn.linear_model import LogisticRegression

def get_logistic_regressor(features, classes):
    # YOUR CODE GOES HERE
    # - Instantiate model with regularization
    # - Fit model to data

    model = LogisticRegression(multi_class = "multinomial",max_iter=1000)
    model.fit(features,classes)

    return model
```

In [33]:

```
model = get_logistic_regressor(xy, c)
preds = model.predict(xy)
accuracy = np.sum(preds == c) / len(c) * 100
print("True Classes:", c)
print(" Predictions:", preds)
print("    Accuracy:", accuracy, r"%")
```

```
True Classes: [0 2 2 2 2 2 0 2 2 2 2 2 0 0 2 0 1 2 0 0 1 1 1 2 0 1 0 1 1 1 0 0
1 1 1 1]
 Predictions: [0 0 2 2 2 2 0 0 2 2 2 2 0 0 0 2 2 2 0 0 1 1 1 1 0 0 0 1 1 1 0 0
1 1 1 1]
    Accuracy: 80.55555555555556 %
```

Plotting Multinomial Classifier Results

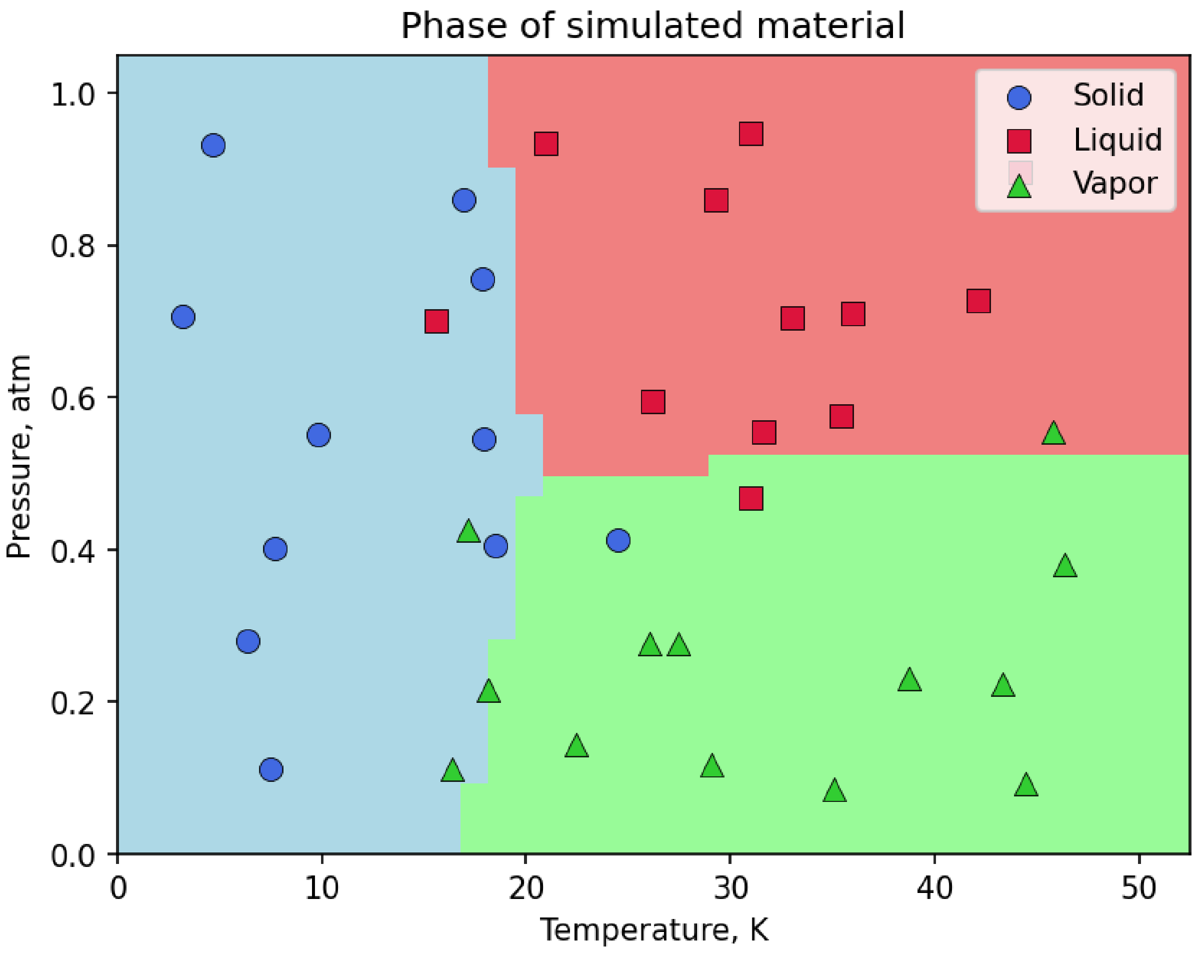
Here, we have made some plotting functions -- run these cells to visualize the decision boundaries.

In [34]:

```
def plot_data(x, y, c, title="Phase of simulated material", newfig=True):
    xlim = [0,52.5]
    ylim = [0,1.05]
    markers = [dict(marker="o", color="royalblue"), dict(marker="s", color="crims
    labels = ["Solid", "Liquid", "Vapor"]

    if newfig:
        plt.figure(dpi=150)

    for i in range(1+max(c)):
        plt.scatter(x[c==i], y[c==i], s=60, **(markers[i]), edgecolor="black", l

    plt.title(title)
    plt.legend(loc="upper right")
    plt.xlim(xlim)
    plt.ylim(ylim)
    plt.xlabel("Temperature, K")
    plt.ylabel("Pressure, atm")
    plt.box(True)

def plot_sklearn_colors(model, res=40):
    xlim = [0,52.5]
    ylim = [0,1.05]
    xvals = np.linspace(*xlim,res)
    yvals = np.linspace(*ylim,res)
    x,y = np.meshgrid(xvals,yvals)
    XY = np.concatenate((x.reshape(-1,1),y.reshape(-1,1)),axis=1)

    color = model.predict(XY).reshape(res,res)

    cmap = ListedColormap(["lightblue","lightcoral","palegreen"])
    plt.pcolor(x, y, color, shading="nearest", zorder=-1, cmap=cmap,vmin=0,vmax=
    return
```

In [35]:

```
plot_data(x,y,c)
plot_sklearn_colors(model)
plt.show()
```



In [ ]: